



链滴

C# 连接基于 Java 开发 IM——Openfire

作者: [tionch](#)

原文链接: <https://ld246.com/article/1540045772026>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

Openfire简介

Openfire 是开源的、基于可扩展通讯和表示协议(XMPP)采用Java编程语言开发的实时协作服务器。Openfire的效率很高，单台服务器可支持上万并发用户。

Server和Client端的通信都用xml文档的形式进行通信。

但是Openfire是Java语言写的，对于C#的dll拓展库相比与java的jar包少的可怜，在网上寻找一番之找到了一个比较好的dll拓展库，agsxmpp是一个专门为C#连接xmpp协议下即时通讯已经搭建xmpp议服务端的dll，同时他有商业版Matrix，博主穷学生一个，没有啥钱去购买商业版，还是采用了通的agsxmpp。

AgsXmpp简介

agsxmpp是AG—Software进行开发的一个开源项目，可以在它的官网进行载源码。

agsxmpp是在2003年开始研发，2008年发布它的最后一个版本，因此它在容性上显然是不很好的。

同时在C#连接Openfire上，agsxmpp中有一个巨坑，加上网上关于agsxm p的开发文档奇少，而且博主没有在官网上找到相关的开发文档（就算有也是全英文看不懂系列），记下开发全过程。

因为agsxmpp并不是专门为Openfire制作的，而是对任何以xmpp协议的时通讯进行连接等服务。如果不对源码进行一定的重写，在某些情况下会出现一些问题。

如果你直接使用 **agsxmpp.dll** 中 **XmppClientConnection** 类进行连接，就算你代码毫无错误，也无法正常连接Openfire，因为

博主只是对源码改了一句话，即可正常连接。

修改 **protocol** 中 **sasl** 下的 **Mechanism.cs** 中源码，将

```
case "DIGEST-MD5":  
    return MechanismType.DIGEST_MD5;
```

注释，因为 openfire 发送数据流 是通过 PLAIN 的，而 agsxmpp 是默认是 通过DIGEST-MD5 送。

同时，在agsxmpp中，还有一个地方表现了对openfire的不兼容，openfire 发送iq节 不接收 to属性，因此还需要修改一个地方

源代码如下

```
public IQ SendIq(agsXMPP.protocol.client.IQ iq, int timeout)  
{  
    synchronousResponse = null;  
    AutoResetEvent are = new AutoResetEvent(false);  
  
    SendIq(iq, new IqCB(SynchronousIqResult), are);  
  
    if (!are.WaitOne(timeout, true))  
    {  
        // Timed out  
        lock (m_grabbing)  
        {  
            if (m_grabbing.ContainsKey(iq.Id))
```

```

        m_grabbing.Remove(iq.Id);
    }
    return null;
}

return synchronousResponse;
}

```

修改后如下

```

public void SendIq(IQ iq, IqCB cb, object cbArg)
{
    // check if the callback is null, in case of wrong usage of this class
    if (cb != null)
    {
        TrackerData td = new TrackerData();
        td.cb = cb;
        td.data = cbArg;
        m_grabbing[iq.Id] = td;

        //iq在agsxmpp中发送Iq节的时候先iq.RemoveAttribute("to")
        iq.RemoveAttribute("to");
    }
    m_connection.Send(iq);
}

```

```

public void SendIq2(IQ iq, IqCB cb, object cbArg)
{
    // check if the callback is null, in case of wrong usage of this class
    if (cb != null)
    {
        TrackerData td = new TrackerData();
        td.cb = cb;
        td.data = cbArg;
        m_grabbing[iq.Id] = td;
        //iq在agsxmpp中发送Iq节的时候先iq.RemoveAttribute("to")
        //iq.RemoveAttribute("to");
    }
    m_connection.Send(iq);
}

```

  登录操作：发送xml消息用 SendIq() 方法

  其他操作：发送xml消息用 SendIq2() 方法

连接上Openfire

官方提供了一个只有三行代码的小型Demo

```

XmppClientConnection xmpp = new XmppClientConnection(server);
xmpp.Open(username,secret);
xmpp.OnLogin+=delegate(object o){xmpp.Send(new Message(JID,MessageType.chat,msg));};

```

我的代码

```

public class XmppLogin
{
    private XmppClientConnection xmppCon;
    private bool isSSL;
    /// <summary>
    /// 是否使用加密连接
    /// </summary>
    public bool IsSSL { get { return isSSL; } set { isSSL = value; } }
    private string userName;
    private string server;
    public string Server { get { return server; } set { server = value; } }
    /// <summary>
    /// 用户名
    /// </summary>
    public string UserName { get { return userName; } set { userName = value; } }
    private string passWord;
    /// <summary>
    /// 密码
    /// </summary>
    public string PassWord { get { return passWord; } set { passWord = value; } }
    private string clientVersion;
    /// <summary>
    /// 客户端版本
    /// </summary>
    public string ClientVersion { get { return clientVersion; } set { clientVersion = value; } }
    /// <summary>
    /// 登录状态
    /// </summary>
    public string LoginState { get { return xmppCon.XmppConnectionState.ToString(); } }
    private int port;
    /// <summary>
    /// 登录端口, 通常是5222, 加密时是5223
    /// </summary>
    public int Port { get { return port; } set { port = value; } }

    public XmppLogin()
    {
        xmppCon = new XmppClientConnection();
    }

    #region 传递一个XmppClient对象
    /// <summary>
    /// 传递一个XmppClient对象
    /// </summary>
    /// <param name="con">需要操作的具体实例</param>
    public XmppLogin(XmppClientConnection con)
    {
        xmppCon = new XmppClientConnection();
        xmppCon = con;
    }
    #endregion

    #region 登录
    /// <summary>

```

```

/// 登录openfire的方法
/// </summary>
/// <returns>返回值为是否登录</returns>
public void Login()
{
    xmppCon.Server = server;
    xmppCon.UseSSL = false;
    xmppCon.Port = 5222;
    xmppCon.AutoResolveConnectServer = true;
    xmppCon.UseCompression = false;
    xmppCon.EnableCapabilities = true;
    xmppCon.ClientVersion = "1.0";
    xmppCon.Capabilities.Node = "http://www.ag-software.de/miniclient/caps";
    xmppCon.DiscoInfo.AddIdentity(new DiscoIdentity("pc", "MyClient", "client"));
    xmppCon.DiscoInfo.AddFeature(new DiscoFeature(agsXMPP.Uri.DISCO_INFO));
    xmppCon.DiscoInfo.AddFeature(new DiscoFeature(agsXMPP.Uri.DISCO_ITEMS));
    xmppCon.DiscoInfo.AddFeature(new DiscoFeature(agsXMPP.Uri.MUC));
    xmppCon.Open(userName,passWord);

    //xmppCon.OnLogin += delegate (object o) { xmppCon.Send(new agsXMPP.protocol.c
ent.Message("testa@118.89.48.159", MessageType.chat, "sdgo")); };
}
#endregion

#region 测试连接
/// <summary>
/// 测试指定的OpenFire服务器和端口是否能连通
/// </summary>
/// <returns>返回是否能连通</returns>
public bool TestPing()
{
    string ipAddress = Server;
    int portNum = port;
    bool CanConnect = false;
    IPAddress ip = IPAddress.Parse(ipAddress);
    try
    {
        IPEndPoint point = new IPEndPoint(ip, portNum);
        using (Socket sock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
rotocolType.Tcp))
        {
            sock.Connect(point);
            CanConnect = sock.Connected;
            sock.Close();
            return CanConnect;
        }
    }
    catch (SocketException e)
    {
        //LOG TODO
        return false;
    }
}
}

```

